Final Project

Logistics: You may choose one of the topics listed below, or choose your own, subject to instructor's approval. Your project must include a written paper and coding in Julia. You may work alone or with a partner; if you work as a pair, you should make only one submission, assigning credit to both partners, and both partners will receive the same grade. You must choose a project topic and a partner (if applicable) by **Friday, November 22**; fill out this form **no later than 11/22**. Your final project is due **Friday, December 6th by 11:59pm**

Format: You will typeset a paper that: summarizes the linear algebra background required for your project; the code you wrote and the results you obtained; and a summary of what you learned. It should be written so that someone else in the class could read and understand your paper. You must include your code—you can typeset it in LaTeX or print a Julia notebook to pdf and include it as an appendix. You should submit a single pdf on gradescope.

Grading: Projects are graded out of 5 points. A project with working code and a readable paper will earn 3-4 points. Projects which also explore some facet of the project beyond the minimal requirements can earn 5 points. Examples include: applying your code to different or larger datasets, implementing variations of your algorithm and testing how they compare, researching and writing up some of the more advanced theory behind your application, or any other extension you are interested in. In other words: take your project in some direction that is interesting to you! Quality is better than quantity: Thoughtful exploration of one or two questions is better than superficial treatment of more. You will be evaluated on:

1. Mathematics and code
    - Does your code produce the intended results on all valid inputs?
    - Is your code well organized and documented?
    - Are the mathematical ideas implemented correctly?
    - Are the choices (e.g. of parameters) made in the code appropriate?
2. Paper
    - Is the paper structured (in sections and paragraphs) to clearly explain your work?
    - Is the writing clear, grammatical, and free of errors?
    - Is the mathematical notation defined and consistent?
    - Are the following things clearly and correctly explained?
        - Mathematical background
        - Algorithms and computations (Pseudocode and examples are critical.)
        - Coding and algorithm design decisions
        - Results and interpretation of computations
    - Does your paper have a complete bibliography that contains the background for your work and any other work you refer to?

Project 1: Iterative methods to compute eigenvectors and singular vectors

Background: In practice, eigenvalues, eigenvectors, singular values, and singular vectors are not computed by finding the roots of the characteristic polynomial (which is the method typically discussed in a linear algebra course). This method is slow and very unstable (roundoff errors can compound to make the calculations inaccurate). Instead, one can use iterative methods. Two such methods are the **Power method** and the **QR method**.

Project: Use the references listed below (and/or additional references of your choosing) to learn what the Power and QR methods are and why they work. Then write some Julia code to compute leading eigenvectors and singular vectors using these methods (you may restrict yourself to symmetric matrices to guarantee that the eigenvalues and eigenvectors are real).

Then you might consider some extensions, such as

- Use deflation or some other method to compute all eigenvectors or singular vectors for a matrix.
- Determine why these methods are fast and accurate and what factors affect the speed of convergence (e.g. ratio of first two eigenvalues). Do some experiments to compare the speed of convergence under different conditions.
- Explore what modifications can be made to make these methods even faster (e.g. shifting, Hessenberg matrices, etc.). Implement some such modifications.
- Alter these methods to handle complex eigenvectors or eigenvalues.
- Explore how eigenvector methods are used in polynomial solvers. (In practice, rather than finding roots of polynomials to find eigenvalues, we do the opposite: Determine eigenvalues to find roots of polynomials.)

References

- Strang (5th Edition) Chapter 11.3 (p. 528-529)
- Foundations of Data Science by Blum, Hopcroft, and Kannan, Chapter 3.7
  (ebook available through CMU library)
- (for advanced topics) Matrix Computations by Golub and Van Loan, e.g. Chapter 7.
  (physically copy available at CMU library)

Project 2: Markov chains, random walks, and PageRank

Background: When you search for a topic on a web browser, you are provided with a list of related websites. How does the web browser decide the order of that list? It uses a page ranking algorithm. For this project you will explore two early page ranking algorithms that rank pages based on the link structure of the internet. The original algorithm used by Google is called PageRank.

Project: Use the references listed below (and/or additional references of your choosing) to learn how PageRank works and implement a version of it on a toy example by hand (an "internet" with just a few nodes and directed links). You should understand and explain the modifications made to change the algorithm from a naive random walk and why this modification is necessary (e.g. with a directed graph, why might a random walk not correspond to a stochastic matrix?). Next, write Julia code that: takes as input a list of websites (indexed by natural numbers) and for each website a list of links; and outputs a list of ranks, one for each site. Apply your code to a few example networks. Finally, implement the HITS algorithm (which is a different page ranking algorithm) on your example networks and compare the results of the two algorithms.

Then you might consider some extensions, such as

- Explore how PageRank and HITS differ; what are the advantages and disadvantages of each algorithm?
- Learn how PageRank and/or HITS can be run efficiently on a truly large graph (e.g. the full internet). What is done to make the algorithms run faster than a naive implementation?
- Learn about and implement successors to or variations of PageRank or HITS.
- Use PageRank or some appropriate variations to analyze a large or interesting network of your choice.

References

- Strang (5th Edition) Chapter 7.3 (p. 386-387)
- https://en.wikipedia.org/wiki/PageRank
- How Google works: Markov chains and eigenvalues, by Rousseau (click for pdf)
- Google's PageRank and Beyond, by Langville and Meyer
(ebook available through CMU library)

Project 3: Spectral Clustering

Background: A clustering problem seeks to group the elements of a data set into a number of groups based on some measure of similarity. Spectral clustering refers to a family of methods using eigenvalues and eigenvectors that in some situations can be more effective than other standard methods. In a spectral clustering algorithm, you should

- Create a similarity graph/similarity matrix for your data using an appropriate measure of similarity.
- Find the first k eigenvectors of the Laplacian or normalized Laplacian matrix, placing them in a matrix A.
- Apply a standard clustering algorithm (such as $k-$means) to the rows (normalized if necessary) of A.
- Interpret the results as clusters for your original data

Project: Use the references listed below (and/or additional references of your choosing) to learn the specifics of spectral clustering. Choose a spectral clustering algorithm, write Julia code that applies it to some data sets, and interpret the results. Discuss what value(s) of k and other parameters you chose and why.

Then you might consider some extensions, such as

- Compare results of different spectral clustering algorithms, or algorithms with different parameters.
- Compare results of spectral clustering with other standard clustering algorithms. Identify characteristics of datasets where each might be superior.
- Apply spectral clustering to some novel data sets.

References

- Spectral clustering by William Fleshman (click for link)
- Spectral clustering for beginners by Amine Aoullayam (click for link)
- Tutorial on spectral clustering by Ulrike von Luxburg (click for pdf)
- Foundations of Data Science by Blum, Hopcroft, and Kannan, Chapter 7 (section 7.5) (ebook available through CMU library)